(12) UK Patent Application (19) GB (11) 2 012 084 A

(21) Application No. **7849864**
(22) Date of filing **22 Dec 1978**
(23) Claims filed **22 Dec 1978**
(30) Priority data
(31) **868146**
(32) **9 Jan 1978**
(33) **United States of America (US)**
(43) Application published **18 Jul 1979**
(51) **INT CL²**
G06F 13/06
(52) Domestic classification
G4A 10P 13E 15A1 15A2 15A3 16D 16J 17P 2E 2F6 MT3
(56) Documents cited
None
(58) Field of search
G4A
(71) Applicants
Honeywell Information Systems Inc.
200 Smith Street, Waltham, Massachusetts 02154, United States of America
(72) Inventors
William E. Woods
Philip E. Stanley
Thomas S. Hirsch
(74) Agents
Lloyd Wise, Bouly & Haig

(54) **Queuing data**

(57) One or more queue structures in a data processing system includes a threaded list of priority frames which are tied to a so-called lock or control frame with synchronization for multiple processing units. Each priority frame includes a priority number, a pointer to the next frame in the list, and data. The lock frame includes a lock, and head and tail pointers pointing to the priority frames at the head and tail of the list. The priority frames are arranged in priority order, highest priority at the head of the list. A new priority frame may be added to the list by a queue-on-head (QOH) or queue-on-tail (QOT) instruction which causes the new priority frame to be added before or after, respectively, the existing priority frames of the same priority. A dequeue-by-address (DQA) instruction causes the priority frame at a specific address to be dequeued, whereas a dequeue-from-head (DQH) instruction causes the first priority frame of a specified priority (or lower priority if no priority frame of the specified priority exists) to be dequeued.

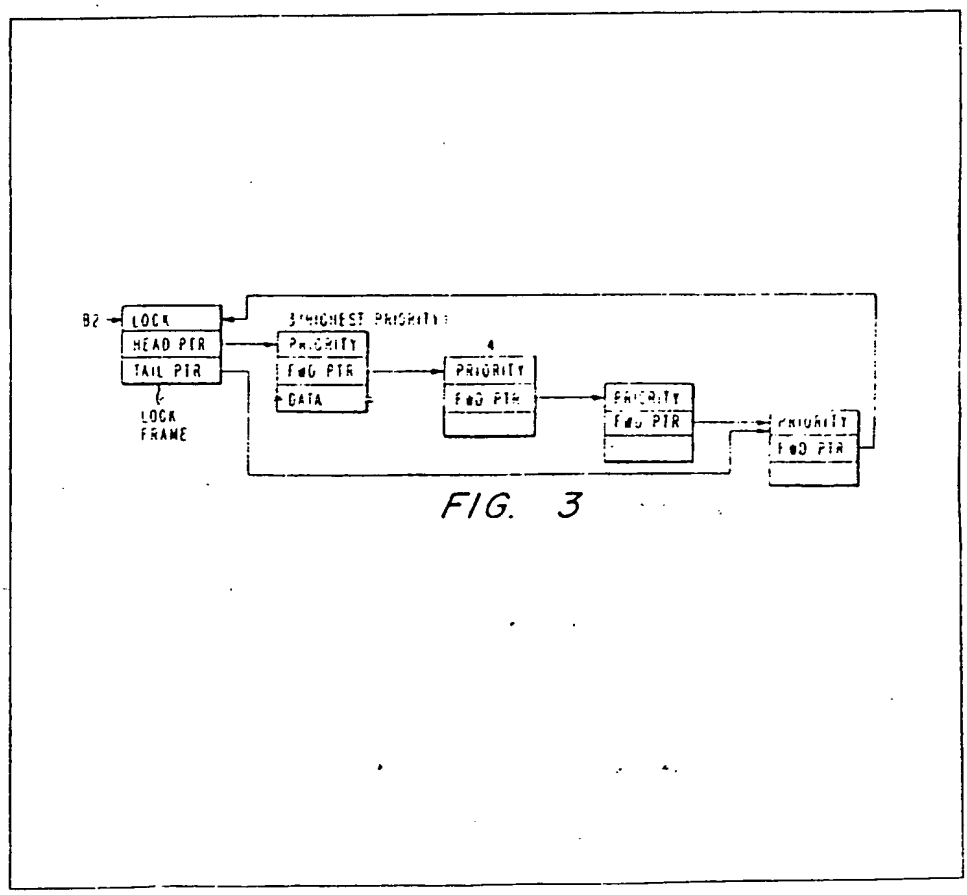

FIG. 3

GB 2 012 084 A

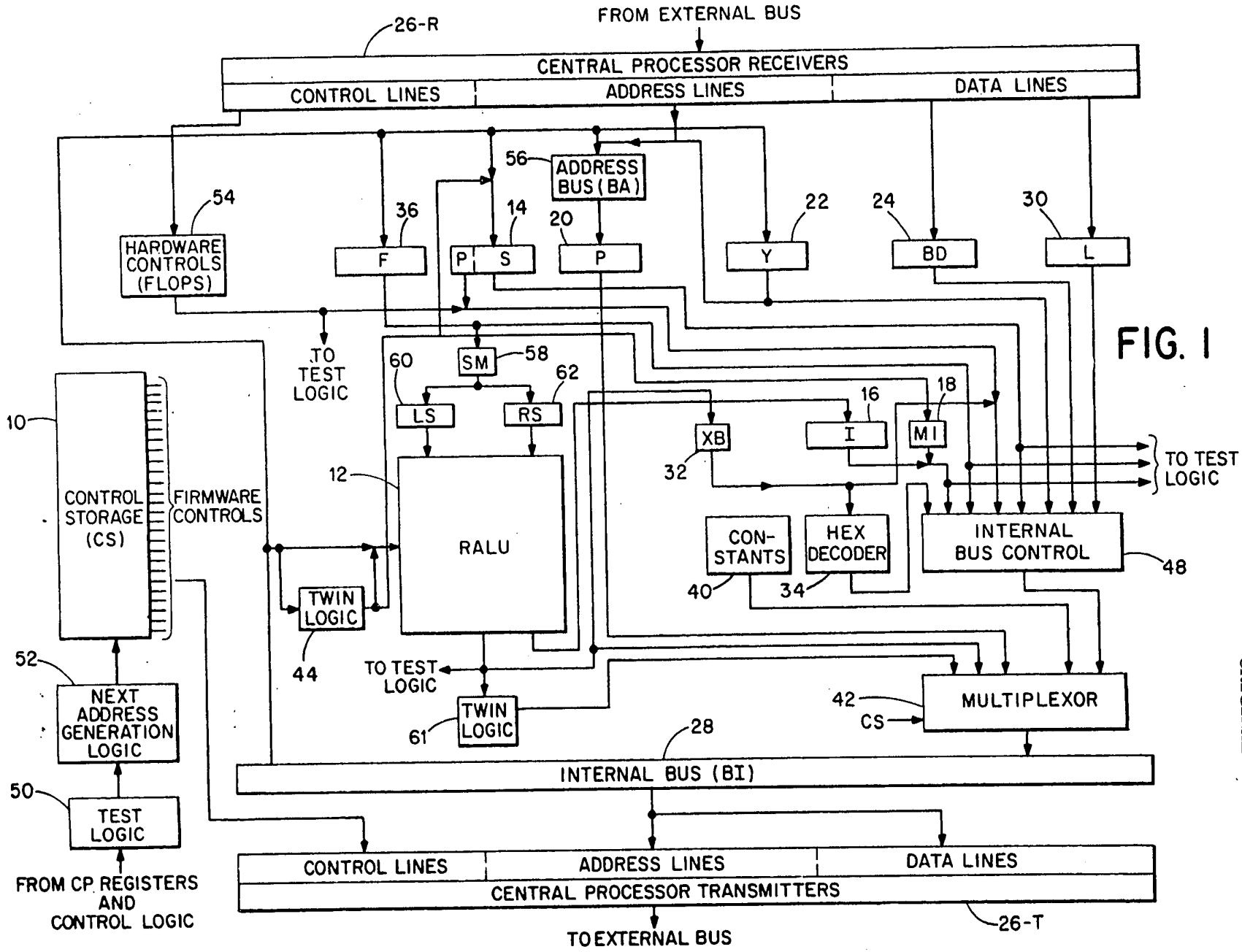

FIG. 1

Formal Drawings
7 Sheets
Sheet 1
ORIGINAL

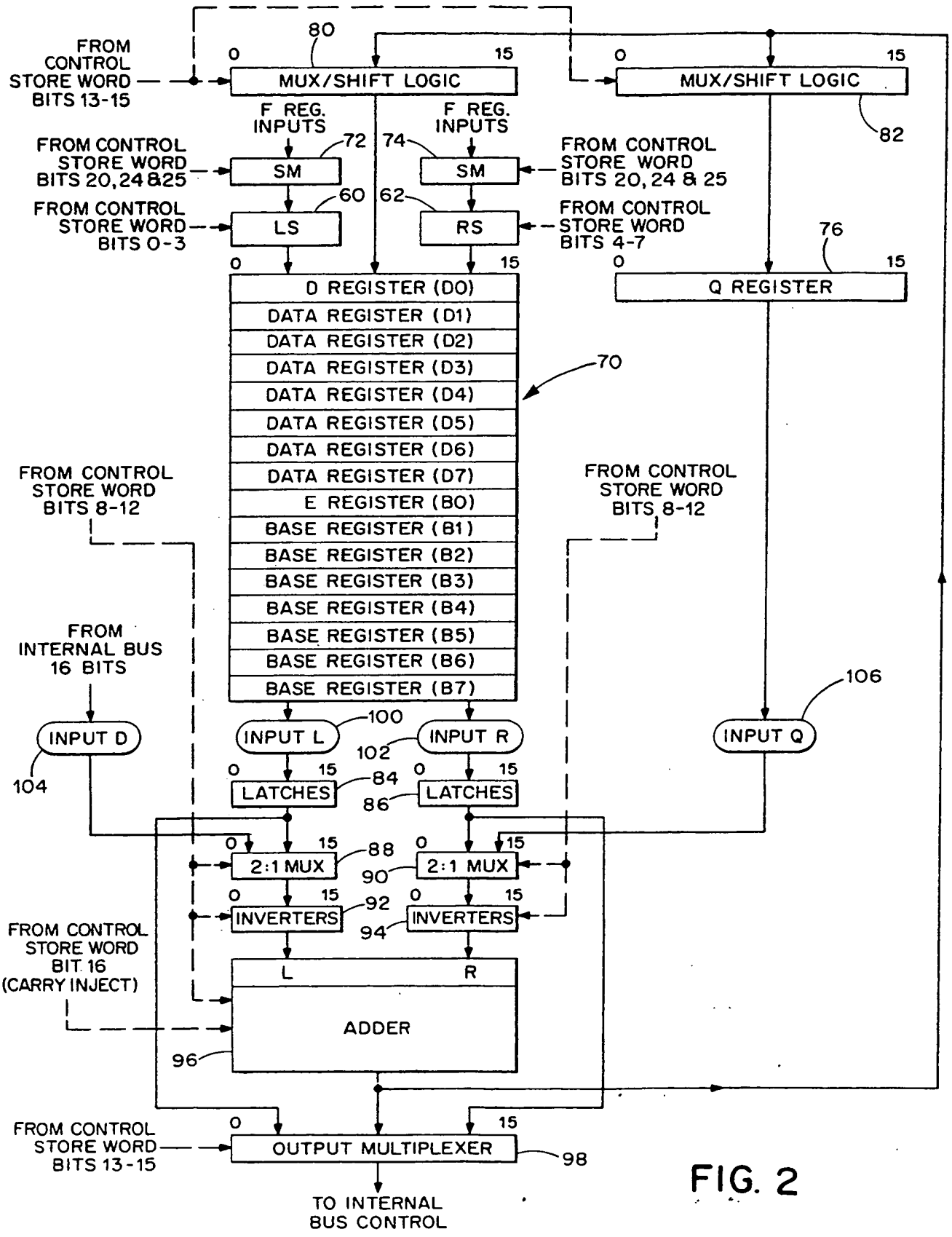

FIG. 2

2012084

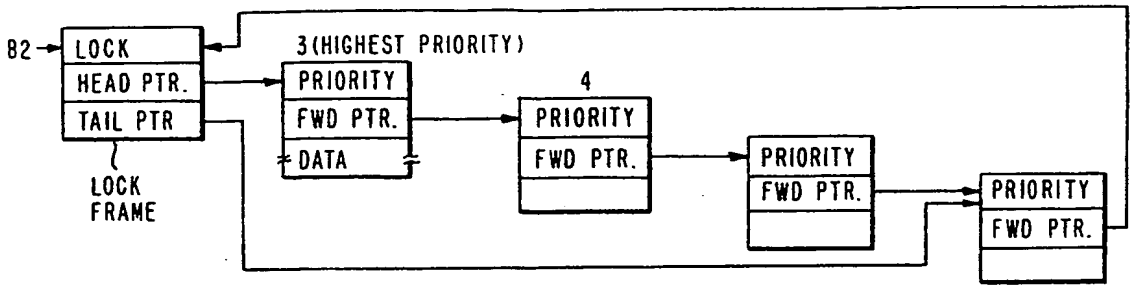

FIG. 3

FIG. 4

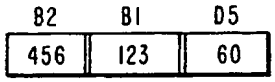

FIG. 5

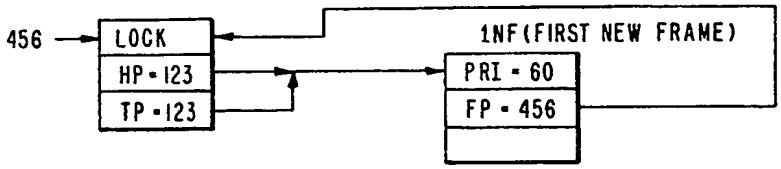

FIG. 6

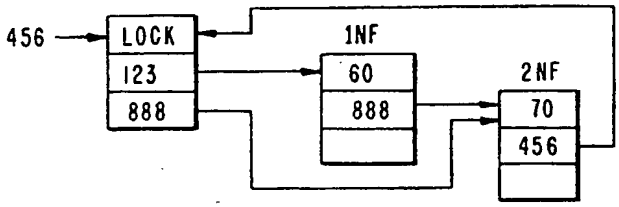

FIG. 7

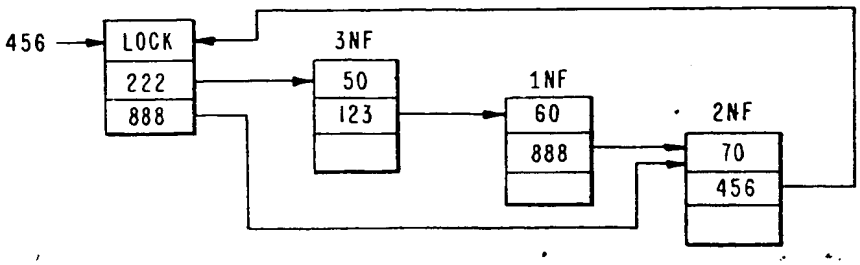

FIG. 8

20120884

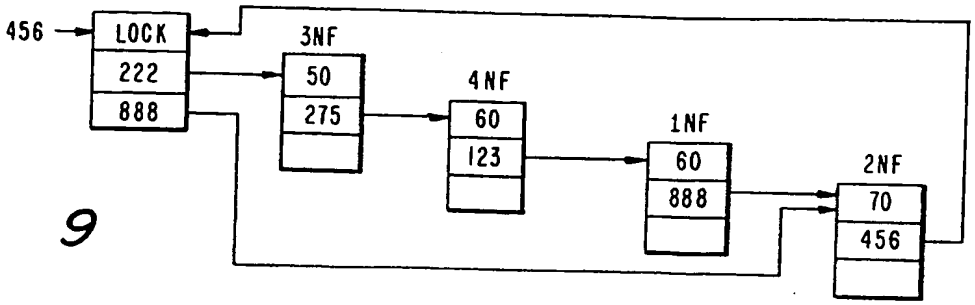

FIG. 9

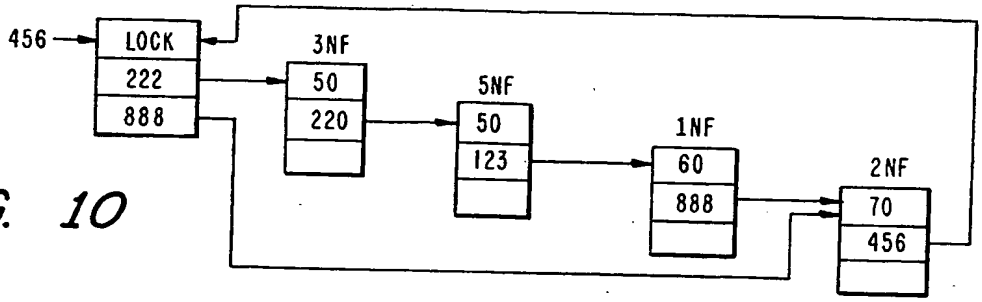

FIG. 10

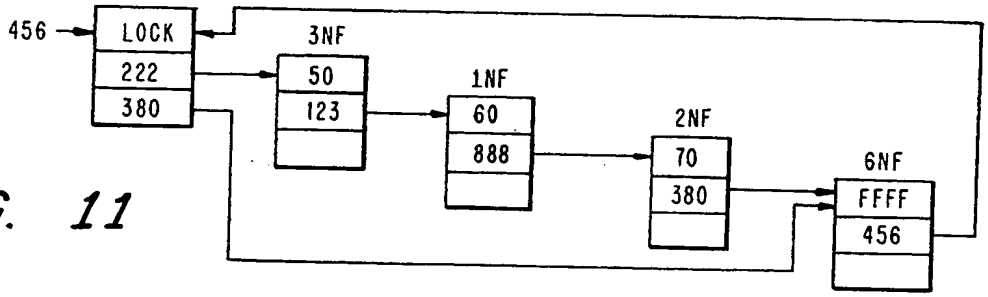

FIG. 11

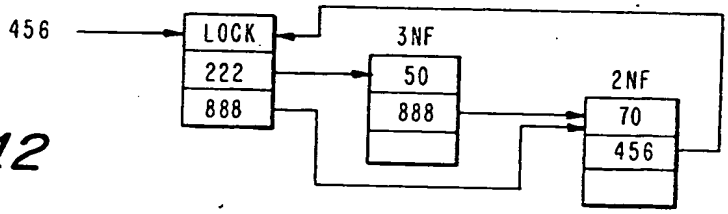

FIG. 12

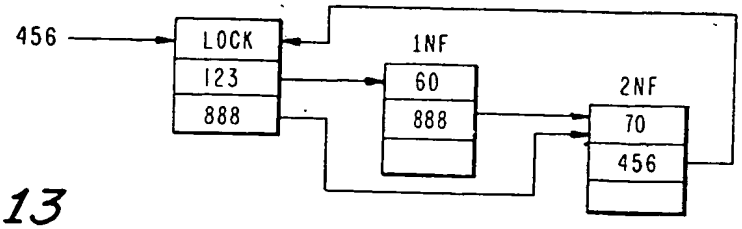

FIG. 13

2012084

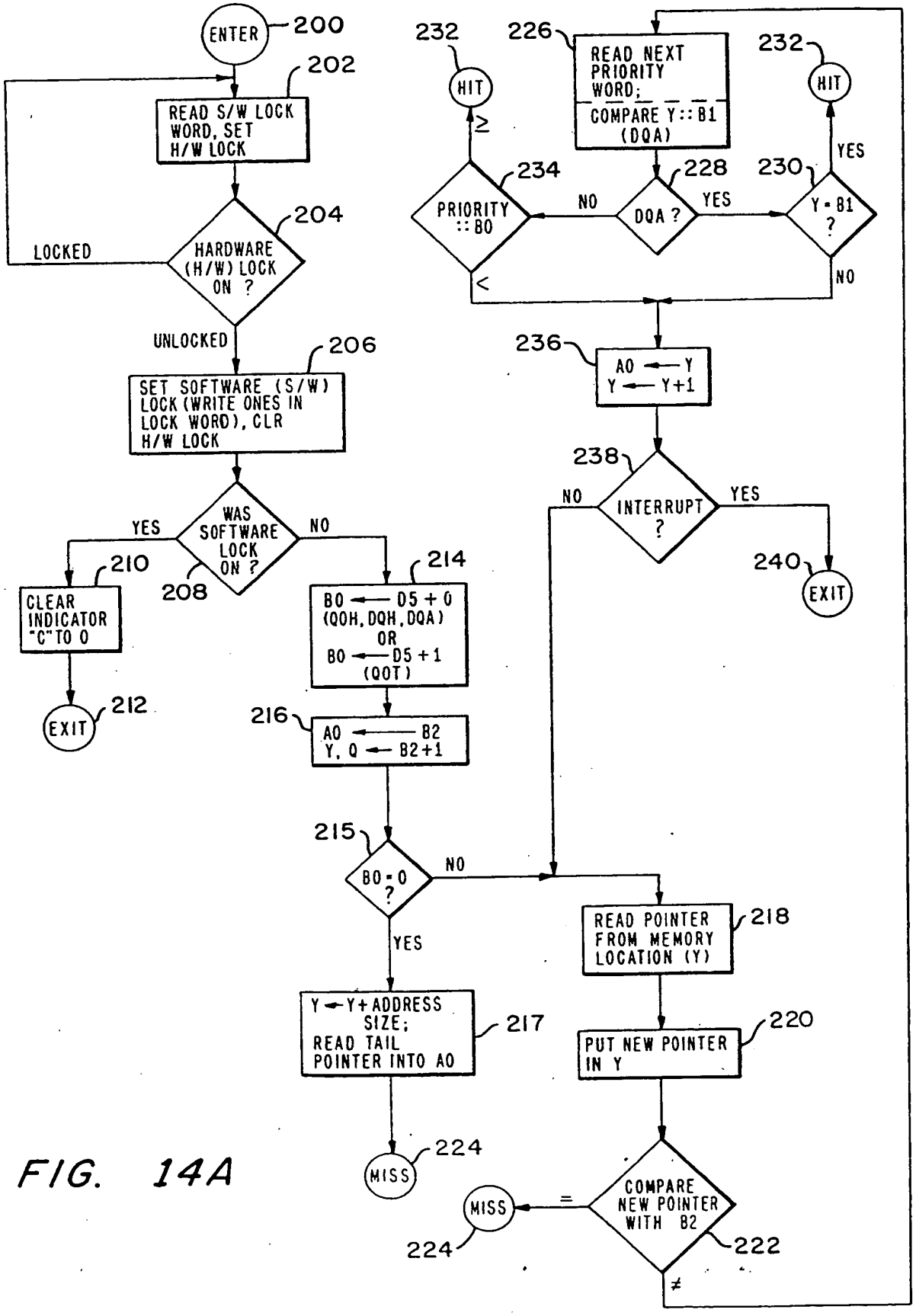

FIG. 14A

2012084

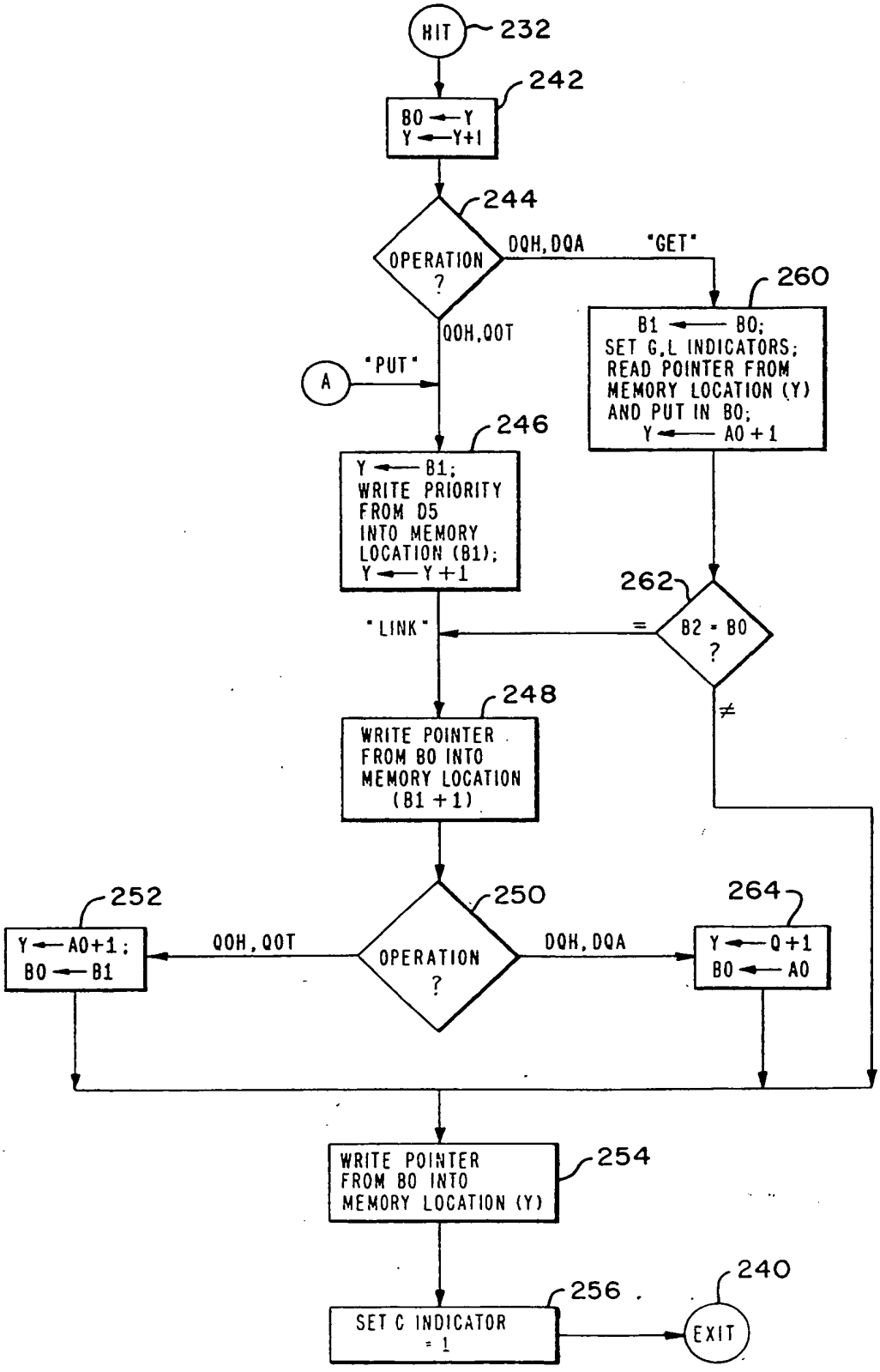

FIG. 14B

2012084

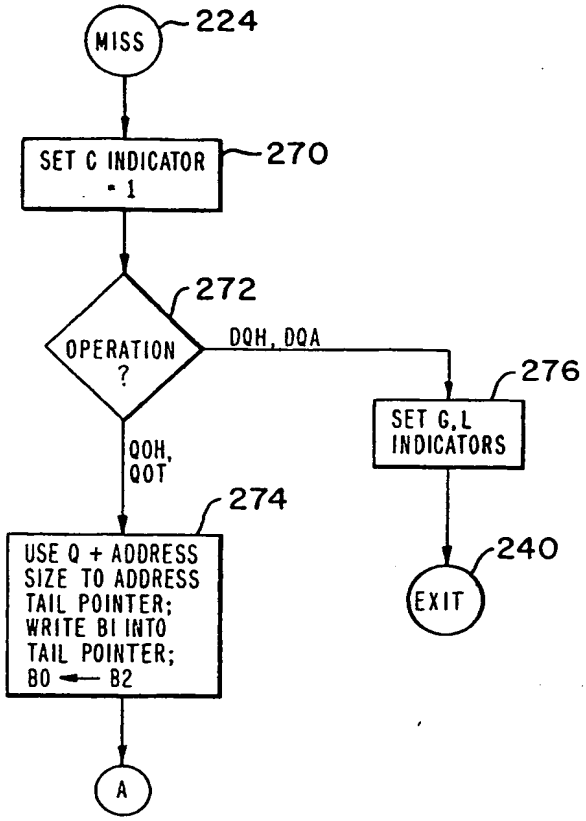

FIG. 14C

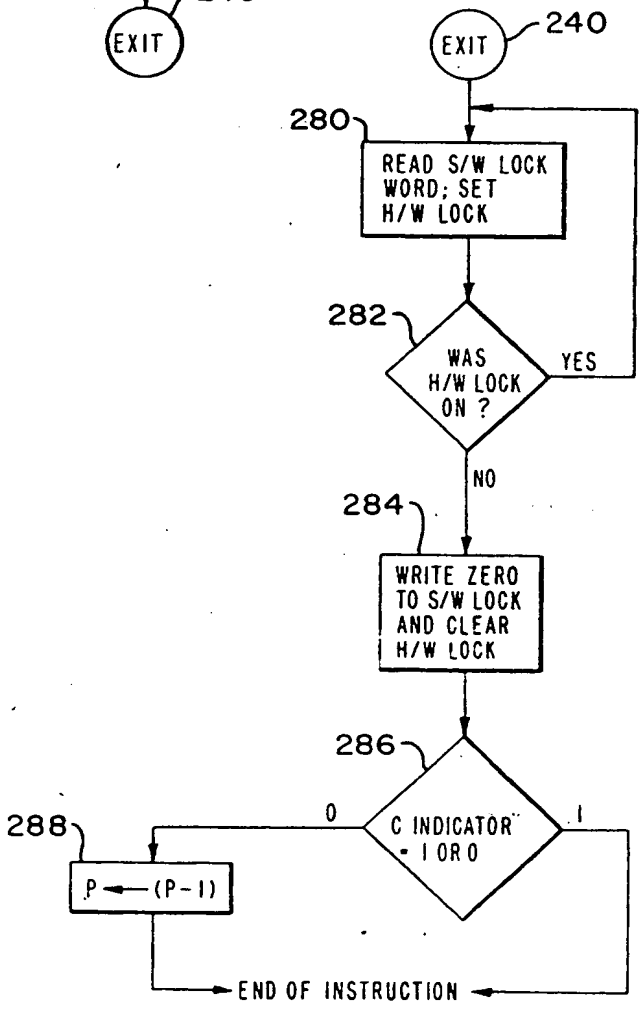

FIG. 14D

2012084

SPECIFICATION

Improvements in or relating to data processing systems

The present invention relates to data processing systems and more particularly to queue structures which may be utilized in such systems.

Queue structures are used primarily for the management of data structures, without the necessity for moving bulk information around in data processing systems, by use of a pointer or header. The queue is used to schedule tasks which are performed on a priority basis. This is done dynamically during processing. Processing is conducted within a task being performed rather than scheduling such processing. Queues are also used in communication buffers for queuing calls. The structures of the present invention may be managed on a last-in, first-out (LIFO) or on a first-in, first-out (FIFO) basis. Queues may also be utilized to maintain a list of free working memory ordered by size or by address. The order of removal of the queues is independent of the order of insertion. In the prior art, it has been known to utilize what has been called an associative memory in order to accommodate the queuing structures. This associative memory technique although theoretically appealing is in actual practice unsuited to handle such queuing tasks due to the relatively high cost and slow access time. In the prior art, as taught in the United States Patent Number 3,449,722, issued June 10, 1969, a new and improved queuing scheme for multiprocessing systems was described in which program requests which cannot be immediately operated on are temporarily stored so that upon the freeing of an originally busy processor section, the oldest queued program request for the particular processor section would be accommodated. Such United States patent suggested the use of a common queue for each such section in the system which could be separately accessed, thereby, by use of the common queue, having established a capability of a string of requests relative to each of the processor sections comprising the data processing system, and thereby avoiding undue expenditure for hardware which is required by such queues. In such queuing structures in data processing systems it is, however, important to provide such queuing structure in a manner such that the queue or frames which are inserted in a list of such queues or frames may be accessed, i.e. enqueued or dequeued, in any manner dependent or independent of the priority of the tasks or subtasks identified by such queues.

It is accordingly a primary object of the present invention to provide an improved queue for use in a data processing system by which the queue structure may be accessed for enqueuing or dequeuing based on the priority of the frames in such queue structure or independent of the priority of such frames, or based on the address of such frames.

According to the present invention, there is provided a data processing system having a queue structure which comprises at least one list of zero or more priority frames coupled with a common control frame, each of said priority frames including a location for a priority number, a location for a first pointer to another one of said priority frames and zero or more locations for including further information associated with such priority frame, and wherein said control frame includes a location for a control word which allows or disallows access to said list of priority frames, a first frame pointer to the first one of said priority frames and a last frame pointer to the last one of said priority frames in said list, and wherein said priority frames may have different priorities as indicated by said priority number, said queue structure further comprising:

A. means for coupling said control frame to said first one of said priority frames by means of said first frame pointer;

B. means coupling said last one of said priority frames to said control frame by means of said first pointer in said last one of said priority frames;

C. means for coupling each of said priority frames, including said first one of said frames and said last one of said frames to a next one of said priority frames by means of said first pointer in each of said priority frames;

D. means for placing new ones of said priority frames in said list in a position determined by said priority number; and

E. means for retrieving one of said priority frames from said list in accordance with, or independent of, said priority number.

In a preferred embodiment of the invention, a data processing system includes a queue structure which comprises at least one list (which may be empty) of priority frames coupled with a common control frame, the priority frame including a location for a priority number, a location for a first pointer to another one of the priority frames and one or more locations for including further information associated with such priority frames. The control frame includes a location for a control word which allows or disallows access to the list of priority frames, and, includes a first frame pointer for pointing to the first one of the priority frames and a last frame pointer for pointing to the last one of the priority frames in such list. Each of the priority frames may have difference priorities associated therewith as indicated by the priority number. The queue structure further comprises apparatus for coupling the priority frames together. The queue structure also comprises apparatus for placing new ones of such priority frames in such list in a position determined by the priority number, and further logic is provided for retrieving one of such priority frames from such list in accordance with or independent of the priority number.

Arrangements according to the invention will now be described by way of example and with reference to the accompanying drawings, in which:–

Figure 1 is a block diagram of a data processing system utilized in conjunction with the queue structure of the present invention;

Figure 2 is a detailed block diagram of the microprocesor which is used if the data processing system of the present invention;

Figures 3 tc 14 illustrate the manner in which the queue structure of the present invention may have frames queued or dequeued; and

Figures 14A to 14D illustrate detailed flow diagrams of the operation of the present invention.

In the queue management system of the present invention, a queue capability is provided that facilitates handling of ordered lists of frames. A frame may include a frame priority number, a next frame pointer, and an associated data structure. Each list is identified by a lock frame which contains a lock word and, in addition, a head pointer and a tail pointer. This is shown in Figure 3. Only one lock frame and associated list of frames is shown in Figure 3, however, it should be understood that more than one lock frame and associated lists of frames may be included in the system. Each threaded list may be associated with a particular task or a number of tasks, thus, these parallel queue structures, each geaded by a lock frame, may be for different tasks or the same task, or for different operations. The queue structure is built for a particular purpose, and when the data processing system indicates that it desires a certain queue, it is usually ready to deal with it, i.e. to operate on the data structure associated therewith. If the system desires to address a different threaded list, then it merely changes the address in a particular one of the registers in the system to point to a new lock frame. The composition of the data structure is not relevant to the present invention, however, it may include data which has been stored away during an operation included in a task, or, in fact, the data structure may be nothing more than another pointer to another location where the actual data is being stored.

There are basically four instructions included in the queue system, two for enqueuing, i.e. placing a frame in the list, and two for dequeuing, i.e. taking a frame from the list. They will be described hereinafter. The lock word is used to insure that only one data processing unit is accessing a particular queue as indicated by a particular lock frame at any given time. Each queue instruction will cause a scan of the frame from the head toward the tail of the list. There is one exception which shall be hereinafter described.

The scan operation will continue until the conditions of the particular command are met by a so-called hit, or until the last frame in the list is reached without a hit, or until an interrupt occurs in the system. When the so-called hit occurs, or if the last frame is reached without a hit, the frame is linked into or out of the list as appropriate, depending upon whether it is an enqueue or dequeue instruction. If an interrupt occurs, the central processing unit will stop the scan and operate on the interrupt condition.

As previously mentioned there are four basic instructions in the queue system of the present invention. They are the queue-on-head instruction (QOH), queue-on-tail instruction (QOT), the dequeue-on-head instruction (DQH) and the dequeue-on-address instruction (DQA). The QOH instruction is used to link a new frame into the list before the first frame that has the same priority number, or before the first frame that has a numerically higher priority number, or as the last frame if no equal or greater priority is found. The QOT instruction links a new frame into the list after the last frame that has the same priority number, or before the first frame that has a numerically higher priority number, or as the last frame if no equal or greater priority is found. If the new frame has the lowest priority that can be given in the system, then the scan operation is bypassed and the new frame is linked after the last frame. The DQH instruction is used to unlink the first frame from the list whose priority value equals or is numerically greater than a priority value indicated by the system. This value is indicated in a specific register as shall be hereinafter described. The DQA instruction unlinks a frame whose priority word address exactly matches an address indicated in the system. This address is indicated in a specific register as shall be described hereinafter.

The operation of the queue structure including the linking and unlinking operation, and including a flow chart illustrating such operation, shall be discussed hereinafter. At this point, the data processing unit architecture in which the queue system of the invention operates shall be discussed.

A block diagram of the data processing system used in the queue system of the present invention is shown in Figure 1. Further details of such data processing system may be found in US. Patent Number 4,047,247, issued September 6, 1977. The system includes a control store 10 which includes by way of example 512 locations, each location including 56 bits. Each such location is capable of storing a firmware word, such firmware words being used to control various hardware operations within the data processor. It is understood that the number of such locations and/or firmware words and the number of bits in such words may be increased or decreased. Operation of a control store and the instruction decoding thereof is shown in the article entitled 'Designing Optimized Microprogrammed Control Sections for Microprocessors', by G.W. Schultz, appearing at page 119 of the April 1974 issue of Computer Design magazine.

Also included in the data processor is a register and logic unit (RALU) 12 which is sometimes referred to as the microprocessor. Figure 2 is a block diagram of the RALU 12 illustrating the details thereof. In general, the RALU is divided into four areas which include a register file, shift logic, arithmetic logic, and control logic. The register file includes data registers, working registers and base registers. The shift logic is used during shift operations and normal transfers of data. The arithmetic logic includes various latches or buffers, multiplexers, inverters and an adder unit. The control logic of the RALU includes selector logic for selecting that portion of data to be operated upon.

The central processor includes various registers some of which are not essential to the present invention, but which will be generally discussed for background purposes. The status/security register 14 contains the system status and security keys. This register includes bit fields which indicate

whether the system is in the privileged state (P) or whether it is in the user state. During the user state, specified instructions will enter a so-called trap routine instead of being executed. The register 14 also includes a field for indicating the I.D. number of the processor, and is set during system configuration. The register 14 also includes a field for indicating the interrupt priority level of the central processor. The current running program in the central processor can be interrupted by any device which requests a level number which is lower than the level number of the running program, wherein the lowest level number indicates a process and/or device which is least interruptable. Such interrupt structure is shown in U.S. Patent No. 4,020,471, issued April 26, 1977.

The indicator register (I) 16 contains the program status indicators. This register 16 includes various fields among which are included fields for indicating the results of any comparison which was made in the system, and indication or status of the last peripheral device which was interrogated, and a field to indicate the state of the last bit tested.

The M1 register 18 contains trap enable mode control keys, which include a field for enabling a trace trap (i.e. a trap which assists in tracing a computer program's operation) for jump and branch instructions.

The program counter (P register) 20 is by way of example a 16 bit register which normally contains the address of the instruction currently being executed. The Y register 22, i.e. the memory address register, is also by way of example a 16 bit register that normally contains the address of data to be accessed in memory. The bus data register (BD) 24 is also by way of example a 16 bit register that receives bus data from the receiver logic 26-R for distribution throughout the processor via the internal bus 28. The interrupt register (L) 30 is also by way of example a 16 bit register that receives a channel number and level of an interrupting device via the receiver logic 26-R.

The XB register 32 is by way of example a four bit register that is used for bit and byte indexing within the processor. The output of this register 32 is coupled to both the internal bus 28 and the decoder logic 34. The instruction register (F) 36 is by way of example a 16 bit register that holds the instruction word as it is received from a memory which may be coupled to the external bus.

The constant generator logic 40 is coupled to provide specific constants to the multiplexer 42 for use in association with the processor's firmware included within control store 10. Decoder logic 34 includes a four to 16 bit multiplexer that is used to generate a mask for bit operations. That is, one out of 16 bits is selected for testing for use by the firmware included in control store 10. The input twin logic 44 provides the capability of either duplicating the most significant (left hand) character (byte) or performing a straight through transfer from the internal bus 28 to the RALU 12. Output twin logic 61 provides similar capabilities.

The internal bus control logic 48, as specified by the firmware word in control store 10, gates the contents of selected processor registers onto the internal bus 28 via the tri-state logic 42. Multiplexer logic 42 includes the logic by which data is transmitted to the internal bus 28, with only one input enabled for transfer at any given time.

Test logic 50 selects by way of example one of 64 possible test conditions, as specified by the firmware word. Depending upon whether the tested condition is true or false, the signal (TSTRUE or $\overline{\text{TSTRUE}}$) is transmitted to the next address generation logic 52. The processor utilizes one of two methods to generate the next firmware address. The first method uses bits of the control store word to form the next address. These bits may for example comprise a 10 bit address field (next address NA) that can directly address one of potentially 1,024 control store locations. The second method obtains the next address from logic circuitry that contains several preassigned addresses. The address selected is determined basically by a decode of the F register 36 contents and the control store 10 outputs.

The internal bus (BI) 28 is by way of example 16 bits wide and is primarily used to transfer data among the processor's registers. Memory addresses and data are also transferred to the external bus via the internal bus 28. The address bus 56 is by way of example 16 bits wide and is used to transfer the addresses for the input and output and memory ready or write cycles to the logic 26–T. The transceiver logic 26 (26R and 26T) include logic circuitry which is the only interface between the central processor and the external bus. All data, address and interrupt signals must pass through the transceiver logic 26. Such transceiver logic 26 as well as the operation of the external bus is described in U.S. Patent No. 4,030,075, issued June 14, 1977.

The select modifier logic (SM) 58 determines which bits of the F register 36 (if any) are used to modify the register file selection performed by the LS and RS fields, i.e. the left select and right select fields of the control store word of control store 10. The SM logic 58 gates F register bits 1 through 3, 10 through 11, 13 through 15, or 12 through 15, depending upon the firmware word, to both the left and right selector logic, i.e. LS logic 60 and RS logic 62. The LS and RS logic use the selector modifier 58 output and the contents of the firmware word for register selection.

The external bus provides a common communication path or interface among all units, including memory, of the system. The external bus is asynchronous in design and units of varying speed are operated efficiently on the system with three types of communication permitted, namely, memory transfers, input/output transfers, and interrupts. The external bus may have coupled thereto, the central processor, a memory unit, peripheral device controllers, communications controllers and the like. The above noted registers, etc. are further described in a Honeywell Information Systems Inc. publication dated January, 1976, entitled, 'Honeywell Level 6 Minicomputer Handbook', order number AS22.

Now referring to Figure 2, the register and logic

unit (RALU) 12 is illustrated in detail. RALU 12 may comprise four model 6701 Microcontrollers manufactured by Monolithic Memories Inc. and described in their publication therefor dated August, 1974. As indicated hereinbefore, the RALU 12 is divided into four basic areas, more particularly a register file, shift logic, arithmetic logic, and control logic. First referring to the register file 70, it includes the data registers D1 through D7, the working registers DO (or D) and E, and base registers B1 through B7. Registers D1 through D7 are by way of example 16 bit word operand registers, with bit zero being considered the most significant bit. Registers D and E are also by way of example 16 bit registers and are used for manipulating data during firmware operations, the register D is sometimes used to hold a copy of the contents of the instruction register (F)36. The base registers are also by way of example 16 bit address registers that can be used for formulating addresses by pointing to any procedure, data or arbitrary location in the system.

Multiplexer shift logic 80 and 82 primarily include two 16 bit multiplexers that are used for both shift operations and normal transfers of data. An additional 16 bit register (Q) 76 is provided for double operand shifts. Data can be shifted left or right by one bit between the multiplexers and any data register within the register file 70. The Q register 76 sometimes contains an unindexed address and the E register (BO) contains an index value.

The arithmetic logic is comprised of two 16 bit latch circuits 84 and 86, two two-to-one multiplexers 88 and 90, two 16 bit inverters 92 and 94, adder unit 96 and an output multiplexer 98. The latches associated with input L 100 receive data from the register file 70 as selected by the left selector logic 60. Similarly, the latches associated within input R 102 receive data from the register file 70 as selected by the right selector logic 62. Outputs from these latches feed both the two-to-one multiplexers 88 and 90 respectively and the output multiplexer 98. The left-hand multiplexer 88 receives data from the internal bus 28 via input D104 and the latches 84 associated with input L 100. The right-hand multiplexer 90 receives data from the Q register 76 via input Q 106 and the latches 86 associated with input R 102. The outputs from these multiplexers are fed through inverters 92 and 94 respectively to the respective L and R inputs of the adder unit 96. The adder unit 96 provides all arithmetic and logical operations. In addition to the L and R inputs, an additional input is provided from control store word bit 16 (carry inject). The adder 96 output is fed to both the output multiplexer 98 and the input multiplexers/shift logic 80 and 82. The output multiplexer 98 is the main output from the RALU 12. Data from the output multiplexer 98 is provided to the internal bus 28 for distribution throughout the processor.

The following is a further discussion with respect to the processor and operation that is depicted in Figures 1 and 2. The central processor is organized around a single internal bus 28 which connects most of the processor elements to each other and to the external bus via receivers 26–R and transmitters 26–T. As indicated hereinbefore, the Y register 22 is the memory address register and the F register 36 is utilized to receive an instruction word during instruction fetches. Various bits on the internal bus 28 are used as inputs to the test logic 50 for use in making firmware branching decisions. The information contained in such various bits from the internal bus 28 can be stored in the test logic 60 and any one of various hardware control flip-flops 54. The internal bus 28 is also an input to the RALU 12.

The internal bus 28 is driven or controlled by several elements including the constant generator 40 which operates under firmware control, the RALU 12, the byte selection register (XB) 32 which is loaded by a shifting from the RALU 12.

The current microinstruction is dynamically available at the output of the control store 10 and is partially decoded with various logical elements and is then used to provide operations with respect to the remaining elements in the system. The next address generator logic 52 utilizes the next address field in the control store word, i.e. the firmware word, and generates a new address dependent thereon and dependent upon test conditions provided by test logic 50. The control store 10 advances to the next address once per processor clock cycle which may be in the order of a few hundred nanoseconds.

Having described the architecture of the data processing system in which the queue system of the present invention may be used, the functional operation of such queue system will now be discussed following which a detailed flow diagram of the queue system of the present invention will be discussed.

By the present invention, the queue frames are arranged in an ordered priority list in increasing order. The threaded list of frames is arranged as shown in Figure 3 with the first frame, called the lock frame, including a lock, a head pointer (or a first frame pointer) and a tail pointer (or a last frame pointer). Each of the frames, of which four are shown by way of example, have a given priority of which priority '3', two '4' priorities and a priority 7 are shown. The head pointer points to the frame having priority 3, 3 to 4, 4 to 4, and 4 to 7, with the priority 7 frame pointing back to the lock frame. The tail pointer of the lock frame points to the priority 7 frame, i.e. the last or tail frame. Thus, as can be seen, there may be frames included which have the same priority.

If a new priority frame is to be added to the queue and if such frame has a priority of 4, this 4 priority frame is inserted either before the two 4 priority frames or after, but not in between. If it is a new priority number which is to be inserted, such as five, then it would be inserted between the last (closest to the tail of the queue structure) 4 priority and the 7 priority frame. The frames are only removed from the head side, i.e. closest to the lock frame, so therefore if a frame having priority 4 is requested from the threaded list of queues then the priority 4 frame closest to the lock frame is taken.

There are four instructions which are provided in order to either enqueue or dequeue frames to and

from the list respectively. They are queue-on-head (QOH), queue-on-tail (QOT), dequeue-from-head (DQH), and dequeue-by-address (DQA). The queues are presumed to be stored in main memory. By way of explanation, and as shown in Figure 4, if there are no frames associated with a particular lock frame, then the head pointer and tail pointer both point back to the lock word of the same lock frame. The base register B2, which is located in the commercially available microprocessor chip as shown in Figure 2, is shown pointing to the lock frame.

As shown in Figure 5, by way of example, the address utilized is indicated to be address 456. In addition to register B2, which has address 456 stored therein, registers B1, also in the microprocessor chip, includes the number address of the frame to be inserted, which is here shown to be 123. The D5 register, also in the microprocessor chip, is used as a priority register, here shown to have a priority of 60 stored therein.

Now with reference to the queue-on-head instruction, the operation thereof is as follows. The firmware on execution of QOH breaks the connections shown for the lock frame without additional frames, as shown in Figure 4. Successively thereafter, the number 123 is stored in the head pointer location, as well as in the tail pointer location of the lock frame. The number 456 is stored in the forward pointer location of the new frame and the priority value 60 is stored in the priority location also of the new frame so that the lock frame and the new frame are now shown coupled as shown in Figure 6.

In further discussion of the operation of the QOH instruction, assume that at a later time, the system assigns the priority number of a second frame to be inserted in the queue structure, to 70, B1 (the address of this new frame) to 888, and assume B2 remains at 456, which is the address of the lock frame, at which time another QOH instruction is executed. Based on this, the queue structure looks as shown in Figure 7. Thus, as can be seen, since the priority equals 70, this is added after the first new frame which has a priority of 60. As can also be seen, the head pointer in the lock frame remains pointing to the first new frame, however, the tail pointer of the lock frame now points to the second new frame. In addition, the forward pointer in the first new frame now points to address 888, the address of the second new frame.

By way of further example, assume that another QOH instruction is executed. This time the priority number in register D5 is equal to 50 with the address of the third new frame, as indicated in register B1, set to be 222, with B2 still having address 456. After execution of this QOH, the lock frame and the three new frames are coupled as shown in Figure 8. As will be noted from Figure 8, the third new frame is coupled between the lock frame and the frame previously inserted with a priority of 60.

Now with reference to Figure 9, assume another QOH instruction is executed, this time with another frame having a priority of 60, whose address is 275, to be inserted in the queue structure. In this case, since it is a QOH instruction, this frame will be inserted before the frame with a priority of 60 which was previously inserted. Accordingly, the queue structure will be changed from Figure 8 to that shown in Figure 9, so that the third new frame has its pointer changed to point to the fourth new frame, with the fourth new frame pointing to the first new frame. Otherwise, the queue structure of Figure 9 resembles that of Figure 8.

The above sequence of operations described the manner in which the queue structure is built up by use of successive QOH instructions. The following will describe two QOT instructions, assuming a base structure existing as shown in Figure 8, and assuming a QOT instruction for inserting a frame having a priority of 50 with an address of 220 as indicated in register B1, and still assuming that register B2 points to the same lock frame with the address 456. Accordingly, the fifth new frame having such priority of 50 will be inserted after the already existing frame having a priority of 50 previously referenced as the third new frame. In such case, the third new frame will have its pointer changed to point to the fifth new frame by replacing the address 123 with the address 220, and the fifth new frame pointer will point by address 123 to the first new frame, it being again noted that the fourth new frame referenced in Figure 9 is not shown in Figure 10 for purpose of ease of illustration only. Thus, in this case, for QOT instruction, the new frame is inserted after the existing frame having the same priority number, whereas in Figure 9, with the QOH instruction, the new frame having the priority number 60 was inserted before the existing frame having the same priority, that is, the new frame was inserted closer to the lock frame in that case (the QOH instruction).

Now with reference to Figure 11, there is a special case in which the QOT instruction may be used where the priority of the frame does not matter. That is, there is a situation which may allow a savings in time in setting up the queue structure, where, in fact, the priority of the new frame does not matter. This instruction is referred to as a QOT instruction having a priority with the highest number, in this case referred to as prior number FFFF (hexidecimal). In this case, such priority is placed at the end of the queue list, after the second new frame in this example, which requires therefore that the pointer of the second new frame point to this sixth new frame with the pointer in such sixth new frame pointing back to the lock word in the lock frame. In addition, the lock frame tail pointer will be changed to point to the sixth new frame, which, in this case, has an address indicated to be 380.

In addition to the QOH and QOT instruction, there are dequeue instructions, and, more particularly, the DQH instruction and DQA instruction. Figure 12 illustrates an example of the DHQ instruction wherein a frame is to be removed. In this case, by way of example, and with a reference to Figure 8 as an existing structure at the time the DQH instruction is received, the DQH instruction indicates that it wants to remove from the queue list a frame whose priority number is 55 or greater, as indicated in register D5. In such case, therefore, the first frame

having a priority equal to or greater than 55 is the first new frame, having a priority of 60. In such case, such frame is removed, the remaining frames, i.e. the third new frame is coupled to point to the second new frame, with the tail pointer of the lock frame, as well as the first frame pointer of the lock frame, remaining the same.

With reference to the DQA instruction, assume that the DQA instruction desires to remove from the queue structure a frame whose address is 222, as indicated in base register B1. In such case, therefore, and with reference to Figure 8, it is the third new frame which will be removed, which, in this case, had a priority of 50, but as explained, this priority does not have any influence on the DQA instruction. Thus, upon removing the third new frame whose address is 222, the queue structure remains as shown in Figure 13, with the first new frame and the second new frame only, with the only change in the lock frame being that the first frame pointer is changed to point to the first new frame rather than the third new frame which has been removed. The first new frame remains pointing to the second new frame, which in turn points back to the lock frame.

Now referring to Figure 14, the operation of the queue system of the present invention will be discussed in further detail by use of flow diagrams. Such flow diagrams depict the operation of the data processing system of Figure 1, and, more particularly, the manner in which the control store words or firmware words are organized and implemented in order to provide the enqueue and dequeue operations in accordance with the four instructions of the queue system of the present invention.

Now referring to Figure 14A, the manner in which the QOH instruction operates and the firmware implementation thereof is shown. At block 200 the QOH instruction is entered following which the address in register B2 of the RALU 12, as more particularly shown in Figure 2, is utilized to address the lock word, which in accordance with block 202 is read and the hardware lock is set. The hardware lock structure is specifically shown and described in United States Patent No. 4,000,485, issued on December 28, 1976, and entitled 'DATA PROCESSING SYSTEM PROVIDING LOCKED OPERATION OF SHARED RESOURCES'. After the operation of block 202 is performed, decision block 204 is entered at which point a decision is made as to whether the hardware lock was on prior to the execution of block 202. If the lock was on, then block 202 is reentered. If the hardware lock was off, the software lock is set by writing binary ONEs into the software lock word as shown in block 206. The interaction of the hardware and software locks is not relevant to the purpose of the present invention, but is used in the implementation illustrated to provide continuous protection of the queue structure in a multiple processor environment. At this point, the hardware lock is cleared. A test is then made as to whether the software lock was on prior to the operation of block 206, as shown in block 208. If the answer to decision block 208 is Yes, then the C-indicator is cleared to a binary ZERO as indicated in block 210, following which there is an exit from this process as indicated in block 212. The C-indicator is included in the I register 16 and is utilized to indicate whether or not the scan has been tried and completed, or whether it was unable to be completed. If the scan is completed, the C-indicator will be set to a binary ONE. If the software lock was not on, then block 214 is entered.

At this point, block 214 indicates that register BO will, depending upon the instruction, be loaded with a certain value. If it is a QOH instruction, then the BO will be loaded with the contents of D5, whereas if it is a QOT instruction, then BO will be loaded with the value of D5 incremented by one. As has been indicated, register D5 contains the priority number and accordingly during the QOT instruction, this priority number is incremented by one before placing it into the BO register. The BO register is utilized as shall be discussed hereinafter.

Block 216 is then entered at which time the contents of register B2 are placed in register AO. AO which is another register which may be found in the microprocessor logic of Figure 2. As will be recalled, register B2 contains the address of the lock word in the lock frame. This address is placed in register AO in order to be able to manipulate such value as shall be seen, without disturbing the contents of register B2. Following this, the Y and Q registers are each loaded with the value of B2 incremented by one. In the next step, as shown in block 215, the question is answered as to whether or not the contents of register BO are equal to ZERO. If the answer is Yes, then block 217 is entered by which the contents of the register are increased in value by the address size following which the tail pointer is placed in register AO. The miss operation is then entered as indicated by block 224. If however the answer to block 215 was a No, then in the next step as shown in block 218, the pointer is read from the memory location indicated by the Y register. This new pointer is then placed in the Y register as indicated in block 220. This memory is a memory which is coupled to the processor of Figure 1 via the external bu as shown in United States Patent No. 4,030,075, issued June 14, 1977.

The new pointer is then compared with a value in register B2 as indicated in block 222. If there is comparison, then this indicates a miss situation as indicated by block 224, which miss situation operates in accordance with the flow diagram in Figure 14c. If there is no comparison, then block 226 is entered at which time the priority word of the next frame is read, following which the contents of the Y register are compared with the contents of the B1 register. This comparison is particularly applicable in the DQA instruction. Thus, the question is then asked as to whether or not this is a DQA instruction, as indicated in block 228. If it is, then the question is asked, as indicated in block 230, as to whether or not the contents of the Y register are equal to the contents of B1 register. If the answer is Yes, then there is a hit situation, as indicated by block 232, and, accordingly, the flow diagram of Figure 14B is entered. If the answer to block 230 is No, then block 236 is entered, as shall be hereinafter described. If the

answer to block 228 was No, i.e. this is not a DQA instruction, then block 234 is entered at which time the value of the priority word is compared with the contents of register BO. If such comparison produces an equal or greater than indication, then the hit situation is indicated, then the operation of Figure 14B proceeds. If such comparison indicates a less than situation that is the value of the priority word is less than the value of the contents of register BO, then block 236 is entered. At this time, the contents of the Y address register are placed in the AO register, following which the Y register is incremented by one. The determination of whether there is an interrupt situation is then made as indicated in block 238. If there is an interrupt situation, then the operation of Figure 14A is exited, as indicated in block 238. If there is an interrupt situation, then the operation of Figure 14A is exited, as indicated by block 240, at which time the operation of Figure 14D proceeds. If there is no interrupt, then, in fact, block 218 is entered again wherein a pointer is read from the memory location indicated by register Y, which, as indicated by block 236, has been incremented by one.

Now referring to Figure 14B, the situation wherein a hit situation is encountered, as indicated in Figure 14A, shall be described. Initially block 242 is entered, at which time the contents of the Y register are placed in register BO and the Y register is incremented by one, following which there is a requirement that the nature of the operation, i.e. the instruction, must be determined as indicated by the decision block 244. If this is a QOH or QOT instruction, then block 246 is entered. It is herein noted that block 246 has another input which is received from Figure 14C during the miss operation. Thus, when block 246 is entered the contents of register B1 are placed in the Y register, following which the priority word is written from register D5 into the memory location indicated by register B1, following which the Y register is incremented by one. Block 248 is then entered, at which time the pointer is written from register BO into the memory location addressed by the value in register B1 incremented by one. The instruction being operated upon is then queried, as indicated by block 250, and if it is a QOH or QOT instruction, block 252 is entered at which time the contents of the AO register are incremented by one and placed in the Y register. The contents of register B1 are then placed in register BO. Block 254 is then entered at which time the pointer is written from register BO into the memory location addressed by register Y. The C-indicator in the I register is then set to a binary ONE as indicated by block 256, which, as indicated hereinbefore, indicates that the scan has been completed. Following this, the operation of Figure 14B is exited as indicated by block 240. The operation then continues as indicated in Figure 14D.

The preceding paragraph has described the operation wherein the instruction has been a QOH or QOT instruction. If the operation is a DQH instruction or a DQA instruction, then the initial decision block with respect thereto, i.e. block 244, branches to the operation indicated in block 260. At this time the contents of register BO are placed in register B1 and the G and L indicators are set in order to, for example, indicate whether the frame was unlinked or whether no match was found, or whether the unlinked frame was the first whose priority equalled the contents of register D5, or whether the unlinked frame was the first whose priority exceeded the value in register D5. Following this, the pointer is read from the memory location indicated by the Y register and is placed in register BO. The contents of the AO register, incremented by one, is placed in register Y. Following the operation of block 260, as indicated in block 262, the question is asked as to whether the contents of register BO are equal to the contents of register B2. Thus, the value of the pointer, which is read from the memory location pointed to by the Y register, is compared with the address of the lock word in the lock frame. If they are equal, then the operation of block 248 is followed, but if not equal, the operation of block 254 is followed. If there was an equal comparison, as indicated by block 262 and this was a DQH or a DQA instruction, then block 250 will branch to block 264 by which the contents of the Q register incremented by one are placed in the Y register, following which the AO register contents will be placed in the BO register. Following this operation, the operation indicated by block 254 is entered.

Having described a situation as indicated in Figure 14B wherein a so-called hit occurs, Figure 14C will now be described for that situation in which a so-called miss occurs as indicated by block 224. Following the entry into this miss operation, block 270 is entered by which the C-indicator is set to a binary ONE. As indicated hereinbefore, the fact that the C-indicator is set to a binary ONE means that the scan has been tried and that the instruction need not be done again. Thus, the operation for that instruction is complete at least for the present time. Following this, the question is then asked in block 272 as to which operation this is. If it is a QOH or QOT instruction, then block 274 is entered at which time the value in the Q register plus the address size (e.g. one or two words) is used to address the tail pointer, following which the contents of the B1 register are written into the tail pointer. Following this, the contents of the B2 register are placed in register BO. At this point, junction A is entered, which junction or point A may be seen as an entry point to block 246 in Figure 14B . The operation then continues as indicated in block 246. If this were a DQH or DQA instruction, as indicated by block 272, then block 276 is entered at which time the G and L indicators are set appropriately. Finally, the operation exits as indicated by block 240.

Now referring to Figure 14D, the manner in which each of such operations exits, as indicated by block 240, shall be discussed. At block 280, the software lock is read and the hardware lock is set. The question is then asked, as indicated in block 282, as to whether or not the hardware lock was on, prior to the setting action indicated in block 280. If the hardware lock was on, as indicated by a Yes answer to block 282, then block 280 is reentered. If the hardware lock was off, as indicated by a No answer,

block 284 is entered whereby binary ZEROs are written into the software lock word and the hardware lock is cleared. The question is then asked, as indicated in block 286, as to whether or not the C-indicator equals a binary ONE or ZERO. If C is equal to a ONE, this means that this is the end of the instruction. If C is equal to a ZERO this indicates the need for decrementing the program counter by one, as indicated in block 288, following which the instruction is also indicated to end.

The manner in which the four generic operations of the queue structure of the present invention operate, has been seen. Use of a lock frame including the lock word therein has also been seen. As has been shown, this lock word, sometimes referred to as the software lock, is checked when starting the operation of any of the four generic instructions, and at such time the lock word is locked, by setting to the appropriate binary value.

However, if it had already been locked, the instruction is exited and is not reentered until the lock word is unlocked, since in fact, by indicating that the lock word is locked, this means that there is another operation being performed with respect to the coupled queue structure, which operation should not be disturbed. When the operation is completed, the lock word is unlocked. Thus, the queue structure is locked, i.e. the lock word indicates that the queue structure should not be disturbed, and is so locked during the scan operation and as the various pointers in the frames are being swapped and/or updated. In general, the hardware lock previously referred to is set and is then unset (the hardware lock is not set during the scan operation) so as to insure that no more than one process being executed in the data processing system may check the software lock at any given mime, thereby avoiding an incorrect answer as to the status of the lock word in the lock frame. There may be more than one software lock for each hardware lock, i.e. there may be more than one threaded list of frames with each having a lock frame and included lock word. By use of such lock words, synchronization for multiprocessing units is provided.

CLAIMS:

1. A data processing system having a queue structure which comprises at least one list of zero or more priority frames coupled with a common control frame, each of said priority frames including a location for a priority number, a location for a first pointer to another one of said priority frames and zero or more locations for including further information associated with such priority frame, and wherein said control frame includes a location for a control word which allows or disallows access to said list of priority frames, a first frame pointer to the first one of said priority frames and a last frame pointer to the last one of said priority frames in said list, and wherein said priority frames may have different priorities as indicated by said priority number, said queue structure further comprising:

A. means for coupling said control frame to said first one of said priority frames by means of said first frame pointer;

B. means coupling said last one of said priority frames to said control frame by means of said first pointer in said last one of said priority frames;

C. means for coupling each of said priority frames, including said first one of said frames and said last one of said frames to a next one of said priority frames by means of said first pointer in each of said priority frames;

D. means for placing new ones of said priority frames in said list in a position determined by said priority number; and

E. means for retrieving one of said priority frames from said list in accordance with, or independent of, said priority number.

2. A queue structure as in claim 1 further comprising means for coupling said control frame to said last one of said priority frames by means of said last frame pointer.

3. A queue structure as in claim 1 wherein said further information which may be associated with said priority frame may include data associated with a task to be performed in accordance with the use of such priority frame or may include a pointer to another location which may contain said information.

4. A queue structure as in claim 1 wherein said means for placing new ones of said priority frames in said list includes means for placing in said list one of said priority frames having a said priority number equal to N, just before, i.e. closest to the end of said list having the control frame, the first said priority frame already placed in said list which has a said priority number equal to or greater than N.

5. A queue structure as in claim 1 wherein said means for placing new ones of said priority frames in said list includes means for placing in said list one of said priority frames having a said priority number equal to N, just after, i.e. closest to the end of said list having said last said priority frame, the said priority frame already placed in said list which also has a priority number equal to N.

6. A queue structure as in claim 1 wherein said means for placing new ones of said priority frames in said list includes means for placing in said list as a new said last one of said priority frames, a said priority frame having the highest said priority number allowed in said system, wherein the higher the priority number the lower the priority of such priority frame for use by said system.

7. A queue structure as in claim 1 wherein said means for retrieving comprises:

A. means for indicating the priority number of a said priority frame to be retrieved; and

B. means, responsive to said means for indicating, for retrieving the priority frame in said list whose said priority number is equal to or if not so equal, then greater than the priority number indicated by said means for indicating.

8. A queue structure as in claim 1 wherein said means for retrieving comprises:

A. means for indicating an address of said priority frame to be retrieved; and

B. means, responsive to said means for indicating, for retrieving the priority frame in said list, without regard to the priority number thereof,

whose address is equal to the address indicated by said means for indicating.

9. A data processing system substantially as described with reference to the accompanying drawings.

Printed for Her Majesty's Stationery Office by Croydon Printing Company Limited, Croydon Surrey, 1979.
Published by the Patent Office, 25 Southampton Buildings, London, WC2A 1AY, from which copies may be obtained.

THIS PAGE BLANK (USPTO)